Building Modern Web Apps with Go

Part 1 of a lot

A Journey not a Destination

About Me

- Lead Developer with CirrusMD
- Skilled iOS developer
- Experienced backend developer
  - *(Ruby on Rails with Postgres)*
- Side project Go developer
- Pretend Android developer
- Hold-a-gun-to-my-head Javascript developer

More about me...

- Serial Startup Engineer
- Type safety and leaning on the compiler
- I ❤️ interfaces
- Patterns and principles not frameworks

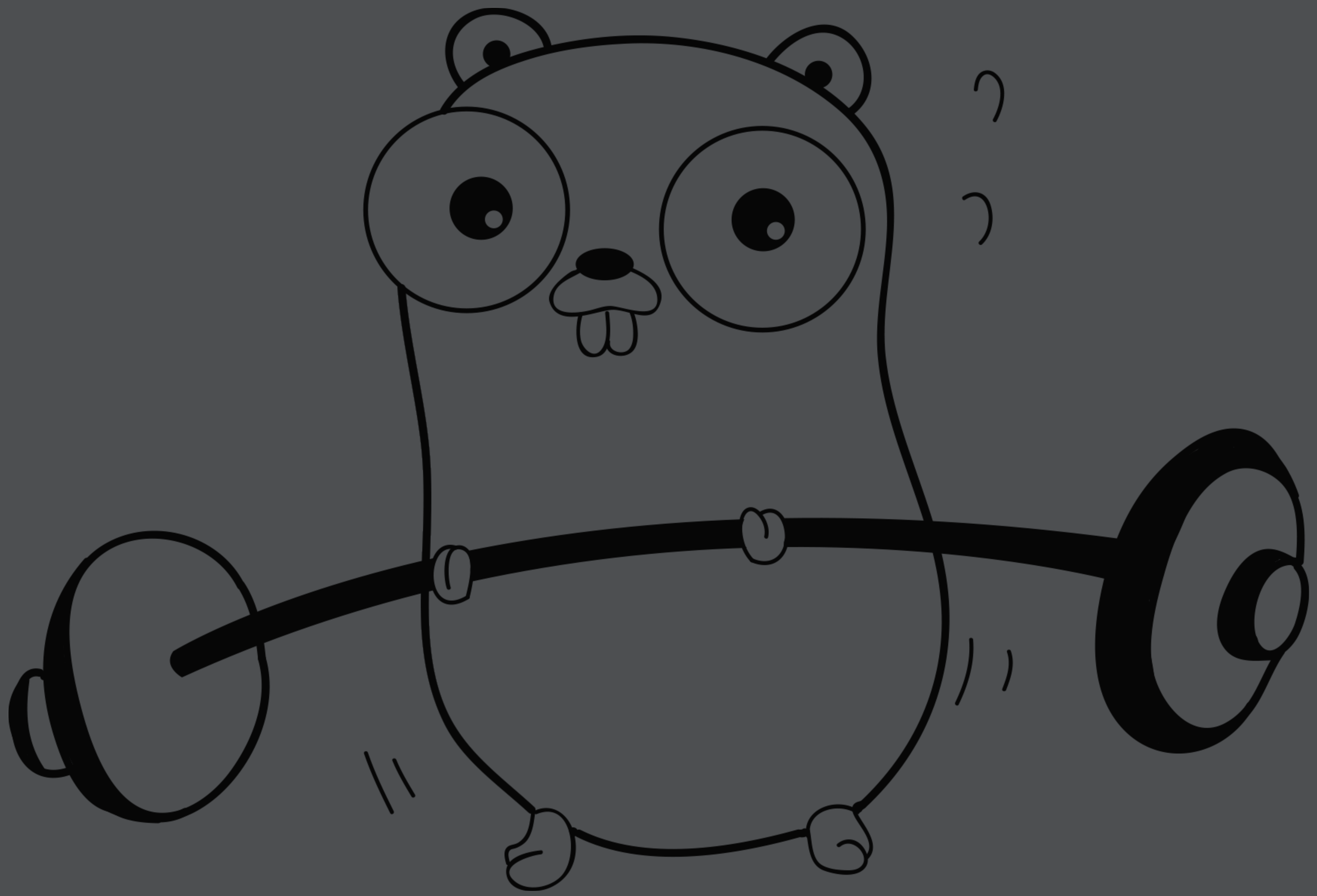
BRACE YOURSELVES
OPINIONS ARE COMING
memegenerator.net

PRIMARY OBJECTIVE

Convince startups to use Go for web apps

Why go with Go?

Go Success Stories



Personal Experiment

Heroku free-tier + hobby PostgreSQL database

Return an array of 1000 1-dimensional JSON objects from database

Result:

50x throughput vs. Rails

Iron.io

30 Ruby on Rails servers to 2 Go servers [1]

[1] http://www.iron.io/blog/2013/03/how-we-went-from-30-servers-to-2-go.html

Parse.com

Push backend 50k connections per node to 1.5 million connections per node[2]

Integration test suite 25 minutes to 2 minutes[2]

Full deploy 30 minutes to 3 minutes[2]

[2] http://blog.parse.com/learn/how-we-moved-our-api-from-ruby-to-go-and-saved-our-sanity/

Malwarebytes

Scaled to 1 million requests per minute while uploading payloads to S3 *without* a worker instance[3]

(ie. no Sidekiq, Delayed Job, Redis, etc. Only Go concurrency primitives)

[3] http://marcio.io/2015/07/handling-1-million-requests-per-minute-with-golang/

Sendgrid

140 Python servers to 1 (souped up) Go Server[4]

[4] Aug 2015 Denver Gophers Meetup

The Free Lunch is Over[5]

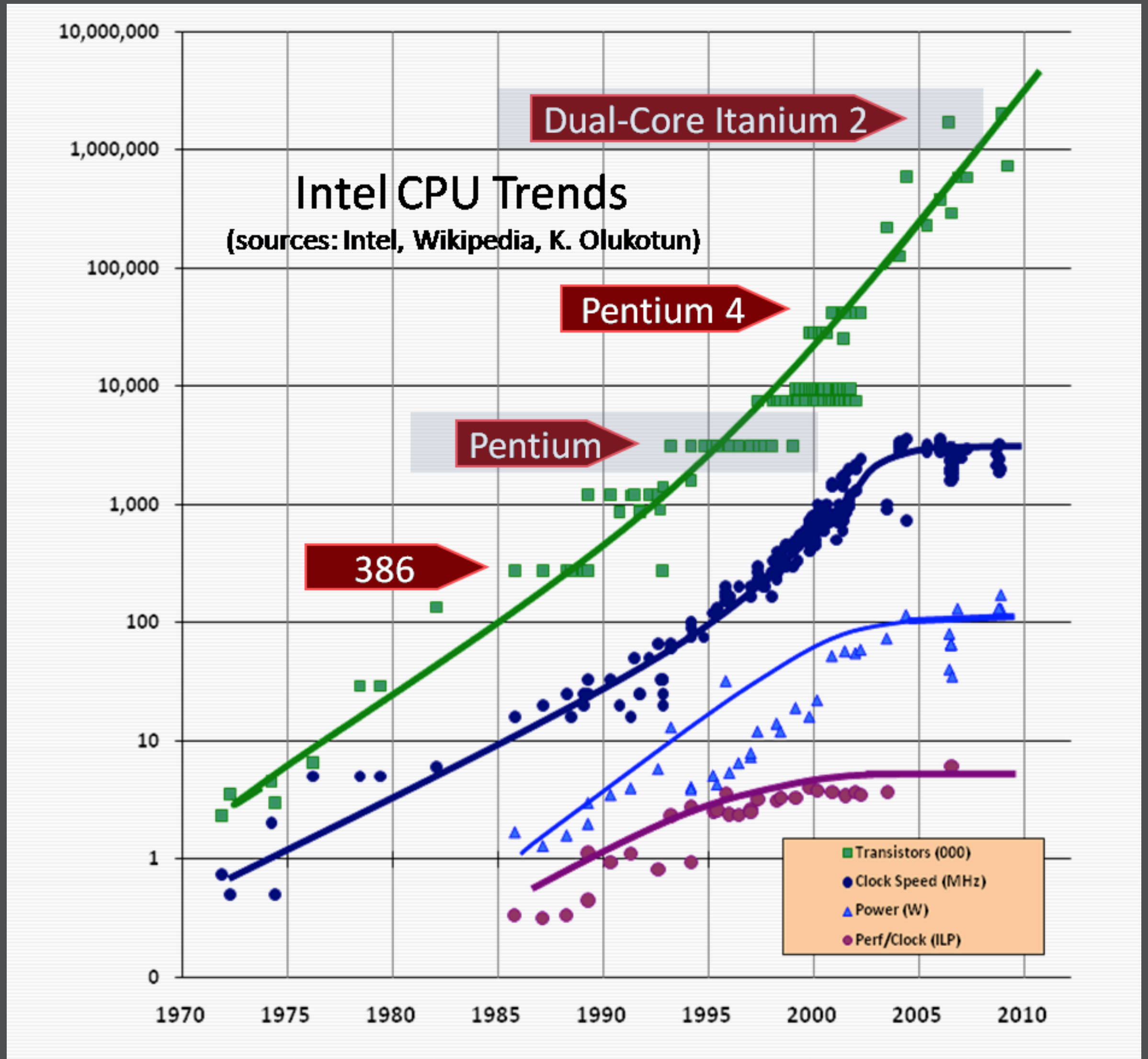


Figure 1: Intel CPU Introductions (graph updated August 2009; article text original from December 2004)

[5] http://www.gotw.ca/publications/concurrency-ddj.htm

Startups can reap these benefits from the beginning.

Let's code!

This is not a modern web app

```go
package main

import "net/http"

func main() {
    http.HandleFunc("/", hello)
    http.ListenAndServe(":8080", nil)
}

func hello(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("hello, world!"))
}
```

What's covered in this talk

Building a Web App in Go

— Project structure

— Routing

— Middleware

Not covered

For future talks

Request Scoped Data, Authentication,
Domain Models, Asset Pipeline,
Templates/Views, Static files,
Graceful shutdown, Transactional Email,

Project Structure

Inspired by Go tools

Feedback encouraged

Don't do this

```
app
├── assets
├── controllers
├── models
└── views
```

WTF does this app do?

Perhaps like this

Demonstrate in Vim

Routing

Straight up plagiarized from Levi Cook's repo:

https://github.com/levicook/foxsays

Middleware

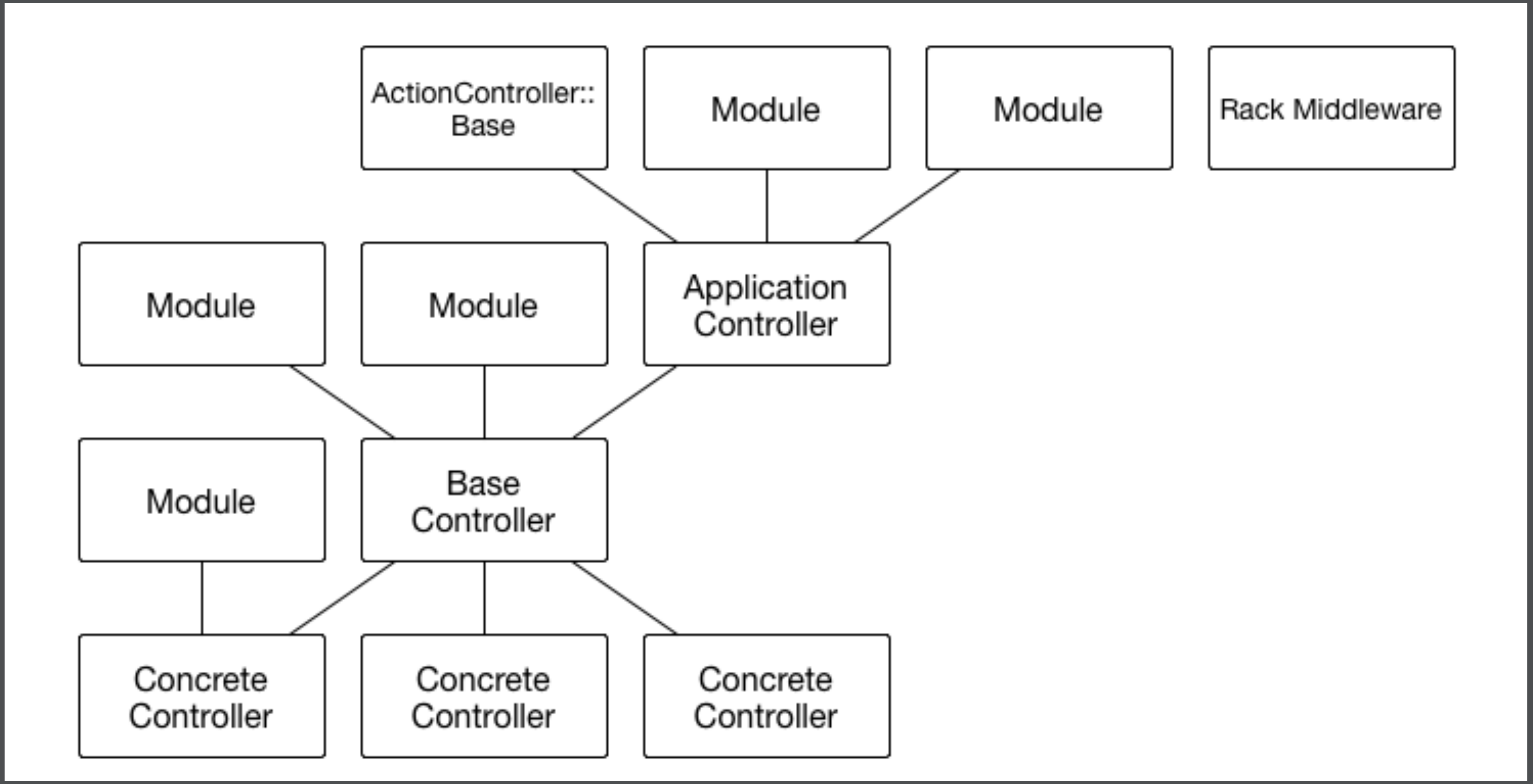
AKA request interceptors

AKA filters

AKA before, after, and around actions

How NOT to do middleware

Anatomy of a "mature" Rails app



```ruby
class SomeController < SomeBaseController
    include Module1
    include Module2

    before_action :do_something_fun  # Middleware is declared here

    # ...
end
```

```ruby
class SomeController < SomeBaseController # Or maybe it's in the base class
    include Module1
    include Module2

    before_action :do_something_fun

    # ...
end
```

```ruby
class SomeController < SomeBaseController
    include Module1 # Er, maybe it's in one of these modules?
    include Module2

    before_action :do_something_fun

    # ...
end
```

Or in the base class's superclass?

Or the superclass's superclass?

Or in a module mixed into the superclass?

Oh actually, it's in application.rb as Rack middleware

```
config.middleware.use 'MySillyMiddleware'
config.middleware.use 'GoodLuckFindingMe'
```

Or in one of several config/environment files...

```
config/environments
├── development.rb
├── production.rb
├── staging.rb
└── test.rb
```

Or in any number of arbitrary initializers

```
Rails.configuration.middleware.insert_before 0, IGuessIAmFirstMiddleWare
Rails.configuration.middleware.use JustShoveMeAnywhereReally
```

Go's Solution

```go
type Middleware func(h http.Handler) http.Handler
```

A function which returns a new handler that wraps the injected handler.

A Refresher

```go
// net/http package
type Handler interface {
        ServeHTTP(ResponseWriter, *Request)
}
```

```go
func MyMiddleware(h http.Handler) http.Handler {
	return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
		// Do stuff here AKA before filter/action
		h.ServeHTTP(w, r)
	})
}
```

```go
func MyMiddleware(h http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
                h.ServeHTTP(w, r)
                 // Do stuff here AKA after filter
        })
}
```

```go
func MyMiddleware(h http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
                // Do stuff here
                h.ServeHTTP(w, r)
                // Then do stuff here aka around filter
        })
}
```

```go
func MyMiddleware(h http.Handler) http.Handler {
	return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
		defer func() {
			// guarantee stuff gets done at the very end
		}()
		h.ServeHTTP(w, r)
	})
}
```

Back to Vim

Third Party Middleware Solutions

- https://github.com/justinas/alice
- https://github.com/codegangsta/negroni

Quest for a Go web framework

- Revel and Beego - too complicated, too much like Rails
- Stop it with controllers. Handlers are just functions.
- Wrong question. Libraries not frameworks.

Other interesting player in the web framework business

Phoenix Framework
Written in Elixer

- Good concurrency, Ruby-like syntax
- But functional and requires Erlang VM

David Nix

Twitter: **@dave_nix**

Github: **https://github.com/DavidNix**

Linkedin***: **https://www.linkedin.com/in/nixdavid**

This talk:

https://github.com/DavidNix/modern-web-apps-in-go

https://github.com/DavidNix/mware

*** Please endorse me for iOS or Go, or anything but Ruby on Rails. I will buy you a beer.